04

Creando el Security Filter

Transcripción

[00:00] Bien. Entonces vamos ya a la parte interesante que es el código. Venimos aquí y ya hemos dicho que lo que vamos a implementar son filtros para conseguir interceptar nuestro request y hacer la lógica de validación del token.

[00:16] ¿Qué es lo primero que tenemos que hacer? Crear el filtro. Entonces vamos a nuestro paquete de infra y aquí en security vamos a darle clic derecho y vamos a crear un nuevo filtro llamado SecurityFilter.

[00:33] Es una clase y bueno, si yo quiero que esta clase sea reconocida por Spring al momento en que la escanee, entonces yo debo usar una anotación. Y ahora la pregunta es para ti. ¿Qué anotación debería usar? Porque esto no es un servicio, no es un repositorio y tampoco es un Controller. Aquí la notación que tendría más sentido component.

[01:05] Component es el estereotipo más genérico de Spring para definir simplemente un componente de Spring. Spring precisa hacer el escaneo en la clase para incluirlo en su contexto. Service, repository y controller son estereotipos basados en componente, explicándolo en otras formas.

[01:24] En Spring todo podría ser Component. Tú podrías nombrar Component una clase de servicio, todo eso, pero para fines de implementar la lectura el código, para un programador Spring divide componente en varios estereotipos para especificar Okay, es un componente pero un tipo en específico.

[01:43] Esto ya es un poco más sobre Springs Corp. cómo funciona pero es muy bueno que lo sepan, dado que nuestro filtro no es un servicio ni un repositorio ni nada, vamos a dejarlo como Component. Entonces paso número 2, si tú ya has visto cómo funciona un filtro, quizás me digas: "Okay, Diego, entonces vamos a extender de filter".

[02:11] No este filter, de Java.xml y bueno, así puede ser, pero nosotros no vamos a usar un filtro directamente de Java, porque estamos usando Spring y Spring ya nos da ciertas funcionalidades, ciertos métodos para que nuestro proceso de validación sea un poco más tranquilo, no tengo que modificar tanto.

[02:34] Para esto entonces yo voy a extender de OncePerRequestFilter, pero yo no puedo extender de OncePerRequestFilter porque es una interfaz, entonces lo que yo voy a hacer aquí es implementar. Y me está dando un error de compilación porque no está implementado método, lo que voy a hacer aquí ya es implement methods, que es un solo método llamado doFilter.

[03:02] En doFilter vemos que recibe tres parámetros. Tienes el filterChain, el response y el request, esto de aquí debe ser conocido para ti si eres familiar con Servlet y filters. Pero bueno, ahora ya tenemos nuestro método doFilter y yo quiero saber primero que todo si este filtro funciona.

[03:25] Entonces yo quiero saber si este filtro funciona. Quiero ver acá. Es una clase abstracta, por eso es que se extiende. Yo estaba en la duda si era una interfaz, pero bueno, si quiero saber si funciona lo que voy a hacer aquí es imprimir algo. Voy a decir "El filtro está siendo llamado".

[03:51] Porque se supone que ya lo implementé. Spring ya lo está escaneando, por lo tanto este filtro debería ser llamado si es que yo entro de un request. Vamos a poner a prueba eso, esperamos a que reinicie mi servidor. Ya debería reiniciar. Quizás ya reinició.

[04:12] Entonces venimos nuevamente a Insomnia, listamos los médicos, enviamos y recibimos un 200 OK y todo muy bien. Pero quiero ver si mi filtro ha sido llamado y no está siendo llamado aún. Voy a detener mi servidor un momento porque no he visto que haya hecho el refresh necesario. Esperamos un poquito.

[04:42] Me está dando, por el pseudocódigo que habíamos escrito y no lo hemos borrado, tiene todo sentido, no estaba comprando mi código. Ahora voy a iniciarlo sin errores. Vemos que inicia mi servicio, inicio de aplicación y vamos a probar el filtro.

[05:10] Le damos enviar y aquí hay dos cosas. La primera me retorna OK, o sea, el request fue excelente, pero ya no recibo nada. ¿Dónde están mis médicos? ¿Dónde está mi lista de médicos? OK. Aquí ya entra la parte aquí el filtro está siendo llamado. Tres veces porque yo clilqueé tres veces, pero esto es bueno porque al menos mi filtro, mi securityFilter ya está siendo usado.

[05:35] Mi securityFilter ya está siendo llamado .Entonces en este momento yo ya puedo hacer la validación del token, pero vamos por partes porque hemos visto que ya no está retornando mis recursos que yo tenía guardados de médico. ¿Esto por qué? Vamos a ver nuestro diseño y en los filtros, si el filtro ejecuta alguna acción, tiene que llamar, sí o sí tiene que llamar al siguiente filtro que va a ser el handling, que va a atrapar este request.

[06:08] Si el filtro no llama al siguiente filtro, es decir si él no hace esa transferencia directamente a la cadena de filtros, recuerden esta definición, cadena de filtros filterChain, si el filtro no hace esa transferencia al siguiente filtro entonces no se devuelve nada, porque todo queda a nivel del filtro.

[06:30] Entonces el filtro necesita saber qué es lo que va a filtrar. Si tú no especificas qué es lo que va a filtrar y a dónde lo va a mandar el filtro dice OK, request fue excelente, te devuelvo tu 200 pero no te devuelvo el recurso porque

no llegas a esta parte de aquí, no llegas a controller. El controller ni siquiera es llamado en este momento, porque el filtro ya lo intercepta.

[06:53] La única forma que tú tienes de hacer eso es usando el parámetro FilterChain, es por eso que les comenté que recuerden eso, el concepto entre filtros, la cadena de filtros, la cadena consiste en que el filtro debe pasarle el request al siguiente, si no, no sucede nada.

[07:12] Para eso lo que yo voy a hacer aquí es comenzar a decirle filterChain.doFilter y tengo aquí mi request y mi response. Esto ya llega de los parámetros aquí. ¿Qué es lo que yo quiero decir con esto? Le digo: "filtro, ejecuta tu magia, ejecuta tu filtro, fíltrame esto, y mándale el request y el response que están llegando del método HTML." Guardamos aquí.

[07:46] Esperemos un momento que reinicie mi servidor y vamos a ver si es que ahora ya está funcionando como debería. Ya reinició el servidor, enviamos nuestro request y listo, ya tenemos nuevamente nuestros recursos y nuestro filtro ya está siendo llamado.

[08:06] Y observen, al inicio, yo solo tenía los tres mensajes de que filtro es llamado y ningún mensaje de Hyibernate, porque como les expliqué antes, no llega, ya este request no llega ni siquiera al Controller si es que el filtro lo hace el doFilter. Esta es la única forma de hacer el filtro, recuerden esto.

[08:27] Si el filtro no llama específicamente al siguiente, no va a suceder y ya está el query de Hybernate. Entonces ya tenemos digamos que el 20% del trabajo listo. Tenemos el filtro, estamos interceptando el request, esto yo lo voy a borrar porque ya sé que está funcionando y también tengo finalmente la llamada al siguiente filtro. Vamos a ver más detalles de implementación en el siguiente video. Nos vemos.